EXPRESS MAIL LABEL NO:
EL707915033US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR PATENT

BLUETOOTH BASEBAND CONTROLLER

Inventor: Min-Shin Ma

BACKGROUND OF THE INVENTION:

This invention relates in general to a Bluetooth baseband controller and, in particular, to an improved system for converting between ACL/SCO data packets and Bluetooth data packets and for retrieving and storing data into buffers in connection with the conversion.

Bluetooth technology is an open specification for wireless communication and networking between personal computers (PCs), mobile telephones, personal digital assistants ("PDA") and other devices. The Bluetooth Specification Version 1.0B is available to members at **http://www.Bluetooth.com/**. This specification is proposed by the Bluetooth SIG, a computer/telecommunications consortium originally consisting of Ericsson, IBM, Intel, Nokia and Toshiba (but now joined by nearly 1,200 other companies, including MicroSoft, Lucent Technologies, Motorola, 3Com and others). This consortium is responsible for defining and promoting the Bluetooth Specification.

According to the Bluetooth Specification 1.0, a Bluetooth system includes a Bluetooth Host, a Bluetooth Host Controller and a physical interface between the host and the host controller. The Host Controller includes a physical bus interface for connection to the Bluetooth Host and HCI firmware portion, a Link Manager ("LM"), a Link Controller ("LC") and a radio frequency ("RF") physical interface. This is illustrated in Fig. 1. In the context of a PC or a PDA, for example, the Bluetooth Host 20 is controlled by the central microprocessor (not shown) and includes a higher layer driver 22, a HCI driver 24 and a physical bus interface 26. So as not to burden the central processor, a separate Bluetooth Host controller 30 is used in order to convert between the HCI level ACL/SCO data packets (hereinbelow referred to as HCI data packets) and the Bluetooth data packets. As shown in Fig. 1, the Bluetooth Host Controller 30 includes a physical bus interface 32 adapted to be connected to the corresponding physical bus interface 26 of the Bluetooth Host 20. Controller 30 also includes an HCI firmware 34,

link manager ("LM") 36, a link controller ("LC") 38 and an RF physical interface (RF PHY) 40.

As specified by the Bluetooth Specification, each Bluetooth device may communicate with a number of other Bluetooth devices, where one Bluetooth device is the master and one or more other Bluetooth devices may be slaves. The link manager 36 in the Bluetooth Host Controller 30 determines what type of connection is to be established with another Bluetooth device through signals transmitted through an antenna 42 and the RF physical interface 40. In other words, the LM determines whether the Bluetooth device including host 20 and controller 30 is to be a master or a slave, and is to communicate with a Bluetooth device which is to be a slave or a master. The LM 36 further determines when the connection is to be established or terminated through physical interface 40 and antenna 42. The link controller 38 controls and maintains the flow of data packets and acknowledgments between controller 30 and another Bluetooth device (not shown). LC 38 provides the handshake required for each type of connections determined by LM 36.

In additional to the above-described functions, LM 36 and LC 38 also perform the function of converting between two different types of data packet formats: HCI data packets and Bluetooth data packets. The mechanism by which such conversion is implemented as specified by Bluetooth Specification 1.0 is described below in reference to Fig. 2. While Fig. 1 shows the functional blocks within the Bluetooth Host 20 and the Bluetooth Host Controller 30, Fig. 2 illustrates in addition, the buffers that are used in a conventional system defined by the Bluetooth Specification 1.0.

As shown in Fig. 2, the data packet conversion occurs between the HCI ACL/SCO buffers 52 and the ACL/SCO buffers 54. The HCI data packets are transmitted through a physical link 56 by the host 20 to the host controller 30 and stored in buffers 52. LM 36 then divides the HCI data packet into segments or portions and sends such portions sequentially through an input/output port 36a of the link manager to input/output port 38a of the link controller through link 58 where each of such portions is sequentially stored in buffers 54 by overwriting the portion that is stored immediately prior to such portion. The packet composer/decomposer 62 then converts each of such portions to the Bluetooth data packet format and transmits each of the converted Bluetooth data packets via the RF physical interface 40 and antenna 42 to another Bluetooth device (not shown).

The HCI data packets include two types of data packets: ACL data packets that contain non-voice type of data and SCO data packets containing only voice data. Some packets may contain both voice and non-voice data; in such event, the voice and non-voice data are separated and processed separately in the same way as those described below. The SCO data packets are transmitted synchronously in order for the voice data to be conveyed in real time so that the rendered voice communication is understood by the listener. If SCO data packets are not successfully transmitted, lost data packets are simply discarded or ignored and no attempt is made to re-transmit them. This is due to the fact that once voice data is lost, there is no reason to re-transmit it since re-transmission would only confuse the listener. Instead the listener would notice immediately that some voice data is lost and ask the speaker to repeat the words spoken, or disconnect voice communication altogether if the problem persists.

The size of the HCI ACL data packet is determined according to the size of buffer 52. The size of buffer 52 depends on the size of the HCI/SCO buffers 46 in the host 20, the HCI physical interfaces 26, 32 and link 56. The number of data bytes in each HCI ACL data packet is usually greater than the maximum number of data bytes in the Bluetooth data packet. Therefore, buffer 52 would need to hold an HCI ACL data packet until it is completely transmitted to the link controller 38 or is flushed.

The HCI ACL data packets are transferred asynchronously through an asynchronous port by the link manager 36 to buffer 54 through link 58. The key consideration in non-voice data transmission is to ensure that all data packets are completely transmitted. For this reason, all data packets that have not been successfully transmitted are re-transmitted until the transmission is successful. The mechanism for transferring ACL data packets will be described in reference to Figs. 3A, 3B, 4 and 5.

Figs. 3A and 3B illustrate, respectively, the HCI ACL data packets and HCI SCO data packets. In reference to Fig. 3A, a HCI ACL data packet ("ACL data packets"), according to the Bluetooth Specification 1.0, includes four fields: a connection handle indicating which connection through which the packet is sent, a PB field indicating whether this packet is the first packet of a Higher Layer Message, a BC field specifying whether the packet will be sent to multiple slaves and a length field indicating the number of bytes in the packet. The HCI SCO data packet of Fig. 3B includes a connection handle and/or field and a single-byte data length field.

Since the number of data bytes in the HCI SCO data packet stored in buffer 52 is not more than the maximum number of permissible data bytes in the Bluetooth data packets, there is no need for the link manager to segment the SCO data packet when stored in buffer 52 or for the buffer 52 to hold the same SCO data packet during multiple transmissions to the link controller 38. In contrast, since the number of data bytes in each ACL data packet can be and is usually more than the maximum number of data bytes permissible in the Bluetooth data packet, the link manager will need to segment the ACL data packet when stored in buffer 52 and buffer 52 will need to hold each ACL data packet until it is completely transmitted or flushed. This is illustrated in reference to Fig. 4.

Fig. 4 is a block diagram showing in more detail the different functional blocks of the Bluetooth Host Controller 30. As shown in Fig. 4, the Link Manager 36 loads the data or payload of the HCI data packet from buffer 52 into buffer 54. The Link Controller 38 then controls the transfer of data from buffer 54 to the packet composer/decomposer 62 for conversion between the HCI data packet format and the Bluetooth data packet format.

The composer/decomposer 62 reads data from the current FIFO (pointed to by switch S1b of Fig. 5A described below) and builds a payload which is appended to the channel access code and the header, thereby forming a complete Bluetooth packet which is subsequently transmitted. The header of the packet indicates whether the payload contains data and/or voice so that the corresponding Bluetooth device may direct the traffic appropriately as indicated by the information in the header. When controller 30 receives a Bluetooth packet, the header of the received packet indicates whether the payload contains data and/or voice. The packet decomposer 62b (see Fig. 5A) will then direct the traffic to the appropriate FIFOs according to similar information in the header of the received packet.

Fig. 5A is a block diagram illustrating in more detail buffer 54 and the packet composer/decomposer 62 of Fig. 4. In reference to Fig. 5A, Link Controller 38 controls switches S1a, S1b, S1c and S1d. For packet transmission, the FIFO that is connected to composer 62a by switch S1b is the current FIFO, and the FIFO that is connected to port 72 by switch S1a is the next FIFO. Similarly, for packet reception, the FIFO that is connected to decomposer 62b by switch S1d is the current FIFO, and the FIFO that is connected to port 74 by switch S1c is the next FIFO. As shown in Fig. 5A, four FIFOs

54txa, 54txb, 54rxa, 54rxb are employed for storing data to be transferred between host controller 30 and a corresponding Bluetooth device (not shown) through the RF physical interface 40 and antenna 42. FIFOs 54txa, 54txb are employed to store data that is to be transmitted from the Bluetooth Host Controller 30 to such corresponding external Bluetooth device. Buffers 54rxa, 54rxb are used to store data that is received from such corresponding Bluetooth external device and stored in controller 30.

The Link Manager 36 loads data from the ACL data packets through an asynchronous port 72 to either one of the two buffers 54txa, 54txb, depending upon which buffer is connected to the asynchronous port 72 through switch S1a. The portion of the ACL data packet that is loaded in a current timer slot would then be transmitted in the next time slot after a complete and successful transmission of data from the other FIFO in the current slot. Ports 72, 74 are some of the input/output ports 38a of Fig 2.

In other words, as illustrated in Fig. 5A, where the positions of switches S1a, S1b are those shown by the solid lines, FIFO 54txa stores the portion of the data that was loaded during a previous time slot and that is to be transmitted to the corresponding Bluetooth device during the current time slot. As illustrated by the solid line position of switch S1b in Fig. 5A, FIFO 54txa is connected to composer 62a. The data stored in buffer 54txa is, therefore, converted into the Bluetooth data format by packet composer 62a and then sent to RF physical interface 40 and transmitted to the corresponding Bluetooth device through antenna 42 as shown in Fig. 1. Prior to the beginning of the same transmit time slot for transmitting the data in FIFO 54txa, Link Controller 38 causes port 72 to be connected to FIFO 54txb through switch S1a in the manner shown in the solid line position of switch S1a in Fig. 5A so that the Link Manager 36 may load into FIFO 54txb the portion of the data from the HCI ACL packet that follows the portion that was loaded and stored in FIFO 54txa.

Assuming that the portion of the ACL data packet stored in FIFO 54txa is successfully transmitted (as indicated by receipt of acknowledgment from the corresponding Bluetooth device), during the next receive time slot, Link Manager 38 causes the switches S1a, S1b to switch, thereby connecting port 72 to FIFO 54txa and FIFO 54txb to packet composer 62a, in the manner shown in dotted lines in Fig. 5A. Then the next portion of the HCI ACL packet stored in FIFO 54txb may then be converted to the Bluetooth data format by composer 62a and transmitted during the next transmit time slot.

According to the Bluetooth Specification 1.0, the Link Controller 38 will not cause switches S1a, S1b to switch until it is confirmed that the portion of the data in FIFO 54txa has been successfully transmitted to the corresponding Bluetooth device. Switches S1a, S1b will not be caused to switch until an acknowledgment has been received by controller 30 from the corresponding Bluetooth device and the data stored in FIFO 54txa has been successfully received by the corresponding Bluetooth device. Upon receipt of the acknowledgment, Link Controller 38 will cause the switches S1a, S1b to switch to the dotted line positions, so that during the next transmit time slot, the data in FIFO 54txb will be transmitted and the Link Manager 36 will load a subsequent fresh portion of data from the HCI ACL data packet to FIFO 54txa through switch S1a in the dotted line position.

FIFOs 54rxa, 54rxb are used to store data that is received from the corresponding Bluetooth device through interface 40 and antenna 42. The Bluetooth data packet received from such Bluetooth corresponding device is decomposed by packet decomposer 62b and the data extracted stored in FIFO 54rxa, thereby overwriting the contents in such FIFO, assuming that the contents of FIFO 54rxa have been unloaded and transferred by the Link Manager 36 to buffer 52 in a prior receive time slot. At the same time, the FIFO 54rxb is connected to an asynchronous port 74 through switch S1c so that the contents of the FIFO can be unloaded by Link Manager 36 and transferred to buffer 52. Assuming that the contents of FIFO 54rxb have been successfully unloaded and transferred by the Link Manager to buffer 52 through port 74 during the present receive time slot, the Link Controller 38 will cause switches S1c, S1d to switch positions so that FIFO 54rxa will now be connected to port 74 and packet decomposer 62b will now be connected to FIFO 54rxb. Thus, during such next receive time slot, the contents of FIFO 54rxa will now be unloaded by the Link Manager and a fresh Bluetooth data packet will be decomposed by decomposer 62b and stored into FIFO 54rxb, thereby overwriting the contents of the FIFO that has previously been successfully unloaded by the Link Manager 36.

A Bluetooth Channel comprises a defined hopping sequence where, for a period of time, the Bluetooth device operates at a particular radio frequency in the 2.4 GHz ISM band. A device operating in a particular hopping sequence will transmit or receive on a particular frequency at a particular moment in time. Bluetooth requires the device to remain on each frequency in its hopping sequence for a period of 625 microseconds.

Each 625-microsecond time period is assigned a number ranging from 0 to 2^{27}-1 and is considered a "slot" in Bluetooth parlance.

Bluetooth devices communicate with each other using a time division duplex (TDD) scheme whereby one device transmits on even slots and the other on odd slots. Transmission of a data packet is guaranteed to start at the beginning of a slot, with different types of packets extending over one or more slots. If a particular type of packet is a multi-slot packet, the device remains at the same frequency until the entire packet has been transmitted. In the next slot, after the transmission of a complete multi-slot packet, the device returns to the frequency required for its particular hopping sequence.

Fig. 6 illustrates an example of time slots according to the TDD scheme. Thus, during the transmit time slot 82, controller 30 may be transmitting a Bluetooth packet to a corresponding Bluetooth device and in the receive time slot 84 that follows slot 82, controller 30 may be receiving a Bluetooth data packet from such corresponding Bluetooth device. The same can be said for subsequent transmit (TX) slots and receive (RX) slots.

As noted above, one of the key concerns in Bluetooth in the data packet transmission is to insure that all data packets are successfully transmitted, even though this may cause delays when a transmission is unsuccessful. For this reason, as is specified by the Bluetooth Specification 1.0, each Bluetooth device that receives Bluetooth data packets would send back an acknowledgment to the sending Bluetooth device to inform the sending device that a data packet has been successfully or unsuccessfully received. If a data packet is successfully received, an acknowledgment ("ACK") is sent. When the receipt of a data packet is unsuccessful, a negative acknowledgment ("NAK") is sent to the sending device.

When the ACK signal is received by the sending Bluetooth device, the sending device is then free to overwrite the FIFO in which is stored the data that was transmitted successfully. When the transmission of a Bluetooth data packet is unsuccessful, the receiving Bluetooth device then sends a negative acknowledgment ("NAK") to the sending device. Upon receipt of the NAK signal, the sending Bluetooth device will not overwrite the FIFO in which the data is stored. The sending Bluetooth device will then attempt to re-send the same Bluetooth data packet during the next TX slot and this process is repeated if necessary until a positive acknowledgment ACK is received from the receiving Bluetooth device.

After the Controller 30 receives a Bluetooth data packet from the corresponding Bluetooth device, the received data packet is decomposed by the Composer 62b to extract the data therein and Link Controller 38 then stores the extracted data into one of the two FIFOs 54rxa or 54rxb. Assuming the positions of switches S1c and S1d shown in Fig. 5A, the data extracted is stored in FIFO 54rxa. After a Bluetooth data packet has been successfully received, the Link Controller then causes a positive acknowledgment ACK signal to be tagged onto the next Bluetooth data packet that is to be transmitted to the corresponding Bluetooth device.

Fig. 5B is a block diagram of a full set of buffers 54 for storing data in connection with communication with a number of slave Bluetooth devices to illustrate a conventional Bluetooth design according to the Bluetooth Specification 1.0. As shown in Fig. 5B, a pair of current and next FIFOs are employed for storing data that is to be transmitted to a particular slave Bluetooth device. According to Bluetooth Specification 1.0, a Bluetooth device acting as a master may transmit data to up to seven slave Bluetooth devices. For this reason, seven transmit buffers, each comprising a current and a next FIFO, are included in buffer 54. These buffers are accessed by the link manager through asynchronous port 72 connected to the seven buffers through a multiplexer 76. Each Bluetooth device acting as the master may communicate with only a single slave Bluetooth device at a time, and multiplexer 76 will select one of the seven buffers for use in storing data that is to be transmitted to its corresponding Bluetooth slave device. A single pair of FIFOs 54rxa, 54rxb are used for storing data packets received from any one of the slave Bluetooth devices. In addition, three pairs of SCO TX buffers are employed for transmitting voice packets to three corresponding slave devices, and three pairs of SCO RX buffers are employed for receiving voice packets from three corresponding slave devices, although again only one of the three TX or RX pairs will be active at any one time. The three pairs of SCO RX buffers are connected to synchronous I/O ports 80 through a multiplexer 78. The three pairs of SCO TX buffers are connected to synchronous I/O ports 80 through a multiplexer 82.

Fig. 6 is a timing diagram illustrating the timing of the transmission and receipt of data packets with reference to one of the TX pairs of FIFOs 54txa, 54txb and one of the RX pairs of FIFOs 54rxa, 54rxb as shown in Fig. 5A between controller 30 and a particular corresponding Bluetooth device. At time t0, the data stored in FIFO 54txa is to be converted and transmitted to the corresponding Bluetooth device. Thus, the TX slot

82 is allocated for the conversion and transmission of such data. Then the next time slot, RX slot 84, is dedicated to the receipt of a Bluetooth data packet from such corresponding Bluetooth device. Thus, if the transmission during the TX slot 82 of the data from FIFO 54txa is successful, an acknowledgment (ACK) that such data has been successfully received will be piggybacked onto the Bluetooth data packet sent by the corresponding Bluetooth device. Upon receipt of such acknowledgment at time t1 by controller 30, the Link Controller 38 is then ready to cause the switches S1a, S1b to switch positions in order to load FIFO 54txa.

During the next transmit time slot 86, the data from FIFO 54txb will be converted and transmitted to the corresponding Bluetooth device. The next receive time slot 88 is dedicated to receipt of the next Bluetooth packet from the corresponding Bluetooth device. If the prior transmission during transmit time slot 86 is successful, the corresponding Bluetooth device will piggy back an ACK signal that is received at time t2. In order for there to be no wasted time slots, this means that between times t1 and t2, or ΔT, the link manager would have to unload the next receive FIFO so that the received data can be stored therein during the next receive time slot, and fill up the next transmit FIFO so that data would be available for transmission during the next transmit time slot. If the link manager is unable to accomplish these tasks within the time period between t1 and t2, one or more transmit and/or receive time slots may be wasted, resulting in lost bandwidth.

According to the Bluetooth specification 1.0, a data packet can occupy up to five slots. The following table gives the data load in terms of the different number of slots:

Category	Packet Type	Data length	Headers (packet & payload)	Total # of bytes transferred in ΔT (TX & RX)	Period (ΔT)	Maximum byte moving time
1-slot packet	AUX1	29 bytes	3 bytes	64 bytes	1.25 ms	19.5 µs/byte
3-slot packet	DH3	183 bytes	4 bytes	374 bytes	3.75ms	10 µs/byte
5-slot packet	DH5	339 bytes	4 bytes	686 bytes	6.25ms	9 µs/byte

As will be seen from the table above, for a single-slot packet transmission, the above transmit and receive operations for transferring 64 bytes of data would both have to occur within 1.25 milliseconds. This means that for single-slot packet transmission, the link manager would have to unload the next receive FIFO so that the received data can be stored therein during the next receive time slot, and fill up the next transmit FIFO during

1.25 milliseconds, in order for there to be full bandwidth utilization. It is found in actual practice, the above-described process is cumbersome and a high percentage of the TX and RX slots are not fully utilized either because of lost data packets or because the Link Manager has not been able to load the transmit buffers and unload the receive buffers on time. The situation is worse for multi-slot packets, since there would be a higher percentage of data compared to overhead to be transmitted, which may result in an even higher loss of bandwidth.

A microprocessor may be used to implement the Link Manager. In such event, the total amount of time delay due to interrupt latency and for moving data should be less than the time period ΔT in order to utilize fully the time slots and, therefore, the bandwidth allocated to Bluetooth data transmission. The interrupt latency is highly dependent upon how many time-bounded tasks and maximum service time of a time-bound task. With the complexity of Bluetooth protocol, it is very difficult for the task of the Link Manager to be completed within the period ΔT using an 8-bit microprocessor.

In conventional Bluetooth devices, the Link Manager 36 and part of the Link Controller 38 are required to load the transmit buffers and unload the receive buffers and to control the switching of the buffers. In such event, the Link Manager and Link Controller may not have adequate speed to carry out the above-described operations within the two TX and RX slots within 1.25 microseconds for single-slot packet transmission. One solution is to use a high power microprocessor, such as a 32-bit microprocessor, for the host controller 30 to speed up the operations. This would, however, be expensive. It is, therefore, desirable to provide an improved configuration for the Bluetooth host controller and/or host in which the above-described difficulties are alleviated.

SUMMARY OF THE INVENTION

This invention is based on the recognition that a key bottleneck in the above-described operations in conventional Bluetooth devices is the need to transmit data between buffers 52 and 54, and that by eliminating this bottleneck, the performance of the Bluetooth baseband controller can be improved even when an inexpensive microprocessor is used for the controller, such as an 8-bit microprocessor.

In the improved configuration proposed by Applicant, buffer 54 is eliminated and the packet composer/decomposer or another control circuit in communication with the composer/decomposer is allowed to access buffer 52 without having to use any other

memory. There is, therefore, no need for the switching mechanism controlled by the Link Controller to switch between the current and the next FIFOs as illustrated in Fig. 5A. Furthermore, there is no need for the Link Manager to segment HCI ACL packets into portions and load these portions into the TX FIFOs 54txa, 54txb and to unload the RX FIFOs 54rxa, 54rxb and re-combine the unloaded data into HCI ACL packets. These tasks are replaced by much simpler tasks so that the operations performed are much simplified. It is therefore more likely for the simplified operations to be performed within the time slot allocated even when an 8-bit microprocessor is used. This results in a higher utilization of bandwidth.

In embodiments where the host and the host controller are combined so that it is possible to convert directly between L2CAP packets and Bluetooth packets without first converting either one into the intermediate HCI level ACL/SCO packets, the invention is applicable as well. In such event, the conversion may be accomplished without having to use any other memory. Where the L2CAP packets are stored in the host memory, only the host memory is involved in the process.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a convention Bluetooth device according to the Bluetooth specification.

Fig. 2 is a block diagram illustrating in more detail the construction of the Bluetooth host and Bluetooth host controller of Fig. 1.

Fig. 3A is a schematic diagram of the packet format of an HCI ACL data packet according to the Bluetooth Specification 1.0.

Fig. 3B is a schematic diagram of the packet format of an HCI SCO data packet according to the Bluetooth Specification 1.0.

Fig. 4 is a block diagram illustrating in more detail the Bluetooth host controller of Fig. 2.

Fig. 5A is a block diagram illustrating in more detail the transmit and receive FIFOs in buffer 54 of Fig. 4.

Fig. 5B is a block diagram of a full set of buffers 54 for storing data in connection with communication with a number of slave Bluetooth devices to illustrate a conventional Bluetooth design.

Fig. 6 is a timing diagram illustrating the operation of the Bluetooth baseband controller of Figs. 2 and 4 in conjunction with the FIFOs in Fig. 5A.

Fig. 7 is a block diagram of a portion of a Bluetooth baseband controller containing an HCI ACL/SCO data buffer, a packet generator and a link controller to illustrate a first embodiment of the invention.

Fig. 8 is a schematic view of the buffer of Fig. 9 and two pointers to illustrate a first embodiment of the invention.

Fig. 9A is a schematic view of a buffer format for transmission of ACL data packets to illustrate a first embodiment of the invention.

Fig. 9B is a schematic view of the buffer format for ACL data packets that are received to illustrate a first embodiment of the invention.

Fig. 10 is a block diagram of a Bluetooth device to illustrate a first embodiment of the invention.

Figs. 11 and 12 are flow charts illustrating a process for control in transmitting and receiving a data packet using the first embodiment of the invention of Fig. 10.

Fig. 13 is a block diagram illustrating the partition between firmware and hardware characteristics of some of the embodiments of this invention.

Fig. 14A is a dataflow diagram illustrating the dataflow for transmitting a data packet using the first embodiment of the invention in Fig. 10.

Fig. 14B is a dataflow diagram illustrating the dataflow when the data packet is received by the device of Fig. 10.

Fig. 15A is a block diagram of a Bluetooth device illustrating an alternative embodiment of the invention.

Fig. 15B is a block diagram illustrating in more detail the device of Fig. 15A to illustrate one implementation of the embodiment of Fig. 15A.

Fig. 15C is a dataflow diagram illustrating a dataflow when the data packet is transmitted using the embodiment of Figs. 15A, 16A.

Fig. 15D is a dataflow diagram illustrating the dataflow when the data packet is received by the Bluetooth device of Figs. 15A, 16A.

Fig. 16A is a block diagram of a Bluetooth device illustrating yet another alternative embodiment of the invention.

Fig. 16B is a block diagram illustrating in more detail and an implementation the embodiment of Fig. 16A.

Fig. 17A is a block diagram of a Bluetooth device illustrating yet another alternative embodiment of the invention.

Fig. 17B is a block diagram illustrating in more detail and an implementation the embodiment of Fig. 17A.

Fig. 17C is a data flow diagram illustrating the dataflow when a data packet is transmitted using the device of Fig. 17A.

Fig. 17D is a data flow diagram illustrating the dataflow when a data packet is received by the device of Fig. 17A.

For simplicity in description, identical components are identified by the same numerals in this application.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Fig. 7 is a block diagram illustrating how the buffer 52 in the Bluetooth baseband controller may be accessed directly by a packet generator to illustrate an embodiment of the invention. As shown in Fig. 7, instead of having to go through the input/output ports 36a, 38a and link 58 as shown in Fig. 2, the packet generator 202 is permitted to access buffer 52, and buffer 54 is eliminated altogether. There is no need for the link manager 236 to load or unload buffer 54. In this manner, one of the important causes of a bottleneck in the conventional Bluetooth device has been removed. Since the packet generator is permitted to access buffer 52 directly, the performance of the Bluetooth baseband controller is much improved and the probability of lost bandwidth is much reduced.

As shown in Fig. 7, the packet generator 202 includes the packet/composer 62 and a packet segmentation/assembly block 204. The composer/decomposer 62 reads data from the FIFO 52 and builds a payload which is appended to the channel access code and the header, thereby forming a complete Bluetooth packet which is subsequently transmitted. The header of the packet indicates whether the payload contains data and/or voice so that the corresponding Bluetooth device may direct the traffic appropriately as indicated by the information in the header. When the Bluetooth baseband controller containing generator 202 receives a Bluetooth packet, the header of the received packet indicates whether the payload contains data and/or voice. The packet decomposer 62b will then direct the traffic to the appropriate FIFOs according to similar information in the header of the received packet.

As shown in Fig. 7, the packet generator includes a packet composer/decomposer 62 described above, and in addition, a packet segmentation/assembly block 204. Since the HCI ACL data packets typically contain more bytes of data than can fit into a Bluetooth data packet, each HCI ACL data packet will need to be segmented into different segments where one or more segments of the data will become the payload of a Bluetooth data packet. The packet composer 62a will build the payload and attache the channel access code and header to become a full Bluetooth data packet ready for transmission. When a Bluetooth data packet is received, the packet decomposer 62b would extract the data from the packet and the packet segmentation/assembly block 204 will store the data extracted at the proper location in buffer 52. The link controller 238 performs functions such as piggy backing the ACK or NAK signals onto the Bluetooth packet generated by generator 202 when the packet is to be transmitted to a corresponding Bluetooth device. The link controller 238 also keeps track of whether an ACK or NAK signal has been received from the corresponding Bluetooth device, where the signal is piggy backed onto a received Bluetooth data packet. It controls a toggle bit in the Bluetooth data packet that is to be transmitted to the corresponding Bluetooth device to indicate whether the packet has been transmitted before in a prior transmission.

Since the segmentation of the HCI ACL data packet and the storing of the extracted data from a received Bluetooth data packet is now performed by assembly block 204 in this embodiment of the invention instead of the link manager, assembly block 204 will need to use a mechanism for keeping track of this process. This is performed in the first embodiment by means of pointers illustrated in Fig. 8. The link manager will convert a data packet from the HCI ACL buffer format such as that shown in Fig. 3A to a new buffer format illustrated in Fig. 9A. As is shown in Fig. 9A, the new format 210 includes two addresses: a previous starting pointer 212 and the next starting pointer 214. The previous starting pointer 212 points to the data byte that was sent in the immediately preceding transmit slot and for which a valid acknowledgment ACK has not yet been received. The next starting pointer 214 points to the first data byte that is to be sent during the very next transmit slot.

The other fields in the buffer format 210 of Fig. 9A are as follows: "BC" which specifies if the packet will be sent to multiple slaves; "PB" which indicates if the packet is the first packet of the Higher Layer Message; "Flow" which is for the HCI level flow control which is included in the payload header; and "na" which is reserved.

Thus, when the link manager first loads buffer 52, it will convert the data format to format 210 of Fig. 9A with pointer 214 pointing to the first data byte (e.g. by pointing to address of data byte 0) that will be sent. The segmentation/assembly block 204 will then fetch an appropriate segment of the HCI ACL data packet using the initial values of these pointers 212, 214 and reset them for the transmission of the next segment. The segment fetched by block 204 is converted into a complete Bluetooth data packet by composer 62a with any acknowledgments piggy backed by controller 238.

The buffer format for buffer 52 for storing data packets received from the corresponding Bluetooth device is illustrated in Fig. 9B. As shown in Fig. 9B, the buffer format 220 includes a data total length field. This information is utilized by block 204 to store the data received at the proper location in buffer 52, in order to assemble a complete HCI ACL data packet from data in a number of Bluetooth packets received over time. By making use of the data total length field of buffer format 220, there is no need for setting or changing pointers when Bluetooth data packets are received from the corresponding device. Thus, after receiving a packet and after the data in the packet has been stored in buffer 52, the generator 202 alters the value in the length field of the buffer, so that data from the next packet will be stored at the right location in the buffer in order to assemble a larger HCI ACL packet.

The overall system diagram of the Bluetooth device according to one embodiment of the invention is, therefore, illustrated in Fig. 10. Thus, instead of the architecture of Fig. 2 where the HCI ACL data packets in buffer 52 would have to be segmented or assembled by the link manager 36 and transmitted or received through link 58 and loaded or unloaded from buffers 54, in the baseband controller 230 of Fig. 10, the packet generator 202 is connected directly to buffer 52 so that it can perform the segmentation and assembly of the HCI ACL data packets in the manner described above.

The operation of baseband controller 230 is described below in reference to Figs. 11 and 12. In reference to Fig. 11, the link manager checks to see whether there is an HCI ACL transmit packet that is pending for transmission (diamond 252). If there is not, controller 230 exits. If there is, the link manager 236 converts the HCI ACL data packet to the new buffer format 210 shown in Fig. 9A (block 254). The packet generator 202 then fetches a segment of the packet using the next starting pointer pointing to the first data byte as the beginning byte of the segment. Any TX ACK/NAK signals are then piggy backed by the link controller 238 to form a complete Bluetooth data packet which

is transmitted to the corresponding Bluetooth device through interface 40 and antenna 42 (block 256). Generator 202 then resets the pointers 212 and 214 by setting the value of pointer 212 equal to that of pointer 214 and by setting pointer 214 to a new value according to the length of the segment that is to be fetched by the generator (block 258). The link manager 236 then checks to see whether buffer 52 is empty (diamond 260). If buffer 52 is empty, then controller 230 exits and if not, controller 230 returns to block 256 to repeat the process until all of the data in buffer 52 has been transmitted. To keep track of whether the buffer is full or empty, the link manager writes values into a register (not shown) accessible by the link controller, so that if the buffer is full when a packet is received, the controller may attache a NAK to the next packet to be transmitted (block 256), and when the buffer is empty during transmission, the system will exit.

In reference to Fig. 12 for the receive routine, generator 202 checks the header of a Bluetooth packet to determine whether the packet is an ACL data packet (diamond 272). If no such packet has been received, that means controller 230 has not received an acknowledgment that the previously transmitted data packet by controller 230 has been successfully received by the corresponding Bluetooth device, so that the previously transmitted data would have to be re-transmitted. For this reason, generator 202 then causes the value of pointer 214 to be set equal to the value of the previous starting pointer 212 (block 274) and controller 230 exits. If an ACL packet has been received, generator 202 checks to see whether a TX ACK is piggy backed onto the packet received. If a positive acknowledgment ACK has not been received or if a negative acknowledgment NAK has been piggy backed onto the received packet, generator 202 again sets the value of pointer 214 equal to the previous starting pointer value (block 274). If a positive acknowledgment ACK is piggy backed onto the received packet, generator 202 checks to see whether there is enough space in buffer 52 for storing data from the received packet (diamond 278). If there is, such data is stored in buffer 52 using the data total length field of format 220 as described above and controller 238 causes a ACK signal to be piggy backed onto the next Bluetooth data packet that is to be sent to the corresponding Bluetooth device in the next TX slot in reference to block 256 (block 280). However, if there is not enough space in buffer 52 for storing the data from the received packet, generator 202 causes link controller 238 to piggy back a negative acknowledgment NAK to the next Bluetooth data packet that is to be sent as described above in reference to

block 256 (block 282). In either case, controller 230 exits after the actions in block 280 or 282.

Where the packet received is a mixed data/voice packet, this is separated by the decomposer 62b, and the data from such packet is processed in the same way as data from a pure ACL packet. The voice portion from such mixed packet is also processed in the same way as voice data from a pure SCO packet. Mixed Data and voice may be processed separately in the same manner as pure ACL and SCO packets and combined by composer 62a into a mixed data/voice packet for transmission.

From the above, it will be evident that a major cause of the bottle neck in conventional systems has been removed. Since the packet generator 202 has direct access to buffer 52, and there is no additional memory that needs to be accessed such as buffer 54, there will be no delay caused by the loading and unloading of such extra buffer as in a conventional Bluetooth device. Since the segmentation and assembly functions previously performed by link manager 36 of the conventional device are now performed by generator 202, the link manager 236 can comprise simpler firmware compared to link manager 36 of the conventional device. As compared to the link controller 38, the link controller 238 does not have to control the switching of buffers 54 and can also be a simpler device compared to the conventional link controller 38. Preferably, generator 202 is implemented in hardware so that the time required for the above-described operations in Figs. 11 and 12 by the generator can be much reduced compared to that performed by firmware in the conventional Bluetooth device.

Fig. 13 illustrates a partition between firmware and hardware in a first embodiment of the invention. As shown in Fig. 13, much of the functions originally performed by the link manager 36 of Fig. 1 is now performed by a reduced link manager 236 and by the packet generator 202 which is preferably implemented in hardware in the preferred embodiments. In the conventional Bluetooth design of Fig. 1, even some of the functions of the link controller 38 are implemented in firmware whereas the link controller 238 of Fig. 13 can be implemented entirely in hardware. Such hardware implementation speeds up the operation of the device and reduces the cost. To illustrate conceptually the difference between the preferred embodiments of the invention and the conventional design, two dotted lines are shown as the partitions between firmware and hardware. As shown in Fig. 13, the partition line 290 is much further to the left compared

to the dotted line 292 of Fig. 1, illustrating the fact that a much greater portion of controller 230 can be implemented in hardware than controller 30.

Figs. 14A and 14B illustrate the dataflow in the first embodiment of Fig. 10. Under the control of host 20, a large L2CAP packet is transferred from the host memory 44 which includes L2CAP buffers through the host controller interfaces 24 and 34 to buffers 52, where interfaces 24, 34 segment the large L2CAP packet into smaller HCI ACL/SCO packets that are stored in buffers 52. Packet generator 202 then performs the segmentation functions described above with respect to the HCI ACL data packets. The segmented data then becomes payload in a Bluetooth data packet generated by generator 202 and transmitted through interface 40. In the receive routine, a Bluetooth data packet is received by interface 40 and directed to the appropriate ACL or SCO buffers by generator 202 and stored at the proper locations in such buffer in the manner described above. The HCI ACL/SCO packets so formed in buffer 52 are then assembled by HCI 24, 34 into large L2CAP packets and stored in buffers 44 for use by host 20.

In the embodiment of Fig. 10, all of the operations of the host 20, and components 22, 24, 26, 44 and 46 in the host, are controlled by a central microprocessor (not shown). Baseband controller 230 is also controlled by a microprocessor (not shown) which, as indicated above, can be an 8-bit microprocessor. Thus, controller 230 relates to host 20 in a similar manner as a computer peripheral relates to a host computer through the physical link 56. The two interfaces 24, 34 perform the function of converting between large L2CAP packets and smaller HCI ACL/SCO packets described above and communication between controller 230 and host 20. Where the functions of host 20 and controller 230 are combined, such as in a cellular phone or personal digital assistant, some of the functional blocks of Fig. 10 can be simplified in the manner shown in Fig. 15A.

As shown in Fig. 15A, the physical link 56 of Fig. 10 is eliminated; the host and the baseband controller are combined into a single device 250 which may be controlled by a single microprocessor or two separate microprocessors. As noted above, HCI 24 and 34 perform the functions of segmentation and assembly and conversion between L2CAP packets and HCI ACL/SCO packets as well as communication between the host and controller 230. By combining the host and the baseband controller, the communication function performed by the two interfaces 24 and 34 can be simplified and the two

interfaces combined into a single interface 234 shown in Fig. 15A, which performs a simplified version of interfaces 24 and 34.

Fig. 15B is a block diagram illustrating some of the functional blocks in one implementation of the system of Fig. 15A. As shown in Fig. 15B, the implementation in such figure differs from that of Fig. 10 in that the buffer for storing HCI ACL/SCO packets is now an embedded memory 52' of the host memory 224. This means that memory 52' may be in the same integrated circuit chip as memory 224. A host memory controller 222a controls the host memory 224. An embedded memory controller 222b controls the embedded memory 52'; controller 222b may also be in the same integrated circuit chip as controller 222a.

A more simplified implementation 260 of the baseband controller than that shown in Figs. 15A and 15B is illustrated in Figs. 16A and 16B. To further simplify the construction, the two memory controllers 222a, 222b are combined into a single memory controller 222' and the two memories 52' and 224 are also combined into a single host memory 224' that includes L2CAP buffers and HCI ACL/SCO buffers. The dataflow process in controller 250 is illustrated in Figs. 15C and 15D. As shown in these figures, for the transmission of packets, the L2CAP packets are stored in the host memory including the L2CAP buffers 224 through host driver 22. They are segmented by HCI 234 into smaller segments and stored in buffers 52' as smaller HCI ACL/SCO packets. These packets are then processed by packet generator 202 and transmitted through interface 40 as described above. In a receive routine, a received Bluetooth data packet is received through interface 40, decomposed and disassembled by packet generator 202 and stored in buffers 52' in order to assemble the larger HCI ACL/SCO packets. These packets are further assembled by HCI 234 into the larger L2CAP packets and stored in host memory 224 and transmitted through driver 22 to the host central processor (not shown).

The dataflow for baseband controller 260 of Figs. 16A and 16B is similar to that for controller 250 shown in Figs. 15C and 15D described above, except that both the larger L2CAP packets and the smaller ACL/SCO packets are stored in a combined host memory 224' in the case of controller 260.

While the role of the link controller 238 of controllers 250, 260 remains essentially the same as that of controller 230 of Fig. 10, the role of the link manager 246 is further reduced compared to the role of link manager 236 of Fig. 10. Since the

physical link 56 has been eliminated, it is now possible for some of the functions of the link manager 236 to be performed by other devices, such as the central processor; the functions that can be transferred includes the determination of master/slave connections and the timing for establishing and terminating such connections. Furthermore, by eliminating the physical link 56, there is less need for communication control.

Controllers 250, 260 can be further simplified into implementation 270 of Figs. 17A, 17B. The dataflow of controller 270 is illustrated in Figs. 17C and 17D. A major difference between controller 270 and the controllers of the other embodiments is that in controller 270, a L2CAP packet is segmented and converted directly into Bluetooth packets whereas in the controllers of the other embodiments, a L2CAP packet is first segmented and converted into HCI level ACL/SCO packets before each of such packets is in turn segmented and converted into Bluetooth packets. Thus the role of the HCI 234" and link manager 246'in controller 270 is further reduced compared to that of HCI 234' of controllers 250, 260. In addition, there is no need for any separation between two buffers: one for storing L2CAP packets, and one for storing HCI ACL/SCO packets. Instead the L2CAP packets are simply stored in buffers that are part of the host memory 274. In controllers 250, 260, the Host CPU (not shown) and host driver 22 segment L2CAP packets into HCI level packets and transmit such packets into the HCI ACL/SCO buffers. In controller 270, since there is no need for such process, the role of the host driver 22' is also reduced compared to driver 22 of the prior controllers 250, 260. The generator 202 may then be used to store data into or retrieve data from the host memory in the L2CAP data format, segment data from such packets into smaller segments for Bluetooth packets, or assemble data from smaller Bluetooth packets into L2CAP data formats in the same manner as that described above.

While the invention has been described above by reference to various embodiments, it will be understood that changes and modifications may be made without departing from the scope of the invention, which is to be defined only by the appended claims and their equivalents.